Apache Spark Word Count App

User

2025-01-13, 01:15:56

Generate a Apache Spark application using Scala language to solve the problem! Given a plain text UTF-8 file URL and a output CSV file URL, write in the output file the occurrence sum of each word in the input file.

Assistant Llama 3.1 70b instruct

2025-01-13, 01:15:56

Here's a simple Apache Spark application in Scala that reads a plain text UTF-8 file, counts the occurrences of each word, and writes the result to a CSV file. ```scala import org.apache.spark.sql.SparkSession object WordCountApp { def main(args: Array[String]) { if (args.length != 2) { println("Usage: WordCountApp ") System.exit(1) } val inputFilePath = args(0) val outputFilePath = args(1) val spark = SparkSession.builder.appName("Word Count App").getOrCreate() val textFile = spark.read.text(inputFilePath) val words = textFile.flatMap(_.value.split("\\s+")) val wordCounts = words.groupBy("value").count() wordCounts.write.csv(outputFilePath) spark.stop() } } ``` To run this application, you need to package it into a JAR file and submit it to a Spark cluster using the `spark-submit` command. Here's an example: ```bash spark-submit --class WordCountApp target/word-count-app-1.0.jar input.txt output.csv ``` This application assumes that the input file is a plain text UTF-8 file and that the output file is a CSV file. The `flatMap` method is used to split each line into words, and the `groupBy` and `count` methods are used to count the occurrences of each word. The result is written to the output file using the `write.csv` method. Note that this application does not handle punctuation next to words. If you want to handle punctuation, you can use a regular expression to split the text into words. For example: ```scala val words = textFile.flatMap(_.value.split("\\b\\w+\\b")) ``` This regular expression splits the text into words that consist of word characters (letters, digits, and underscores) bounded by word boundaries.